Vérification de contrats logiciels à l'aide de transformations de modèles Application à Kmelia

Pascal Andre, Gilles Ardourel, Mohamed Messabihi

Vérification de contrats logiciels à l'aide de transformations de modèles

Application à Kmelia

Pascal André — Gilles Ardourel — Mohamed Messabihi

LINA UMR CNRS 6241
Université de Nantes
{Prénom.Nom}@univ-nantes.fr

RÉSUMÉ. *Un modèle de système logiciel couvre des aspects orthogonaux (structure, fonctions, dynamique). Vérifier formellement des propriétés de ce modèle implique donc d'utiliser plusieurs outils et techniques. Ces outils sont basés sur des modèles spécifiques, écrits dans des langages adaptés aux aspects traités. La transformation de modèle apparaît donc comme une solution naturelle pour la vérification de systèmes. Nous illustrons dans cet article l'utilisation conjointe de modèles spécifiques et des transformations associées pour vérifier les propriétés relatives aux contrats. Les contrats sont déclinés en différents niveaux sur les éléments de modélisation. La mise en œuvre est réalisée avec des plugins Eclipse dans COSTO, une plate-forme dédiée au modèle à composants* Kmelia.

ABSTRACT. *A software system model covers orthogonal aspects (structure, functions and dynamics). Formally verifying properties of such a model requires the use of various techniques and tools. These tools are based on specific models, written in languages that are specific to the target aspects. Model transformation appears to be the adequate technique to tackle the challenge. In this paper we illustrate a way to use specific models and the related transformations to check a set of various properties covered by contracts. Different levels of contracts are spread over the model elements. This work is implemented using Eclipse plugins in COSTO, a platform dedicated to the* Kmelia *component model.*

MOTS-CLÉS : *Composant logiciel, services, contrats, langages spécifiques, vérification de propriétés, transformations.*

KEYWORDS: *Software component, services, contracts, specific languages, property verification, transformations.*

1. Introduction

Partant d'un modèle abstrait il est important d'en déterminer les propriétés avant de le raffiner par transformations successives vers un modèle opérationnel ou le code final. Cet aspect devient crucial lorsque le système doit respecter des contraintes de sûreté de fonctionnement. Nous avions mis en place les bases d'un processus de vérification de cohérence de diagrammes UML dans (André *et al.,* 2005). Nous poursuivons ce travail sur des modèles à composants et services sûrs (Meyer, 2003) en visant la vérification des propriétés du système modélisé (non redondance de données, atteignabilité, sûreté, vivacité, *etc*). De plus, nous nous inscrivons dans la lignée de la conception par contrat.

Les modèles couvrent souvent des aspects transversaux (structure, fonctions, dynamique). De fait, il n'existe pas d'outils universels permettant de fonder une vérification unifiée des propriétés. L'idée est alors de se tourner vers plusieurs outils adaptés aux aspects identifiés (*theorem proving* ou *model checking*), chacun assurant une partie de la vérification. Les outils ayant en général un modèle et un langage spécifique, on se trouve ici dans une problématique de transformations de modèles (extraction, fusion, injection...). Nous illustrons dans cet article l'utilisation conjointe de modèles spécifiques et les transformations associées pour vérifier les propriétés relatives aux contrats. Les contrats sont déclinés sur les différents éléments de modélisation. La mise en œuvre est réalisée avec des plugins Eclipse dans COSTO (*COmponent Study Toolbox*), une plate-forme dédiée au modèle à composants Kmelia (Messabihi *et al.,* 2010). Nous exposons dans la section 2 l'extension d'un modèle abstrait à composants par un modèle de contrats et de vérifications. La section 3 présente les niveaux de contrats et leurs éléments. La section 4 décrit une mise en œuvre. La section 5 résume la contribution et trace des perspectives.

2. Modèle à composants et services

Dans cette section, nous survolons les concepts principaux d'un modèle abstrait à composants (CM) auxquels sont rattachés les contrats (CCM) et les vérifications (VCCM). La partie *CM* de la figure 1 est inspirée du métamodèle Econet[1]. Les classes communes telle que *ModelElement* ne figurent pas ici. Les composants sont caractérisés par un état et des services (ou opérations) définis dans leur type. Ils sont assemblés sur leur points d'accès (*EndPoint*) qui sont des ports, des interfaces ou des services selon le modèle considéré. Le détail d'un service varie aussi de la signature seule jusqu'au comportement détaillé des communications en passant par la composition de services. Un assemblage (ou architecture) relie les composants, éventuellement via d'autres assemblages, sur une relation de clientèle (contrat client/serveur classique). Un composite les encapsule dans une relation d'inclusion (contrat parent/enfant). Le comportement et les communications sont décrits, selon le langage, par des structures de contrôle, des expressions régulières, des automates, *etc*.

1. http://www.lina.sciences.univ-nantes.fr/coloss/projects/econet/

Dans le modèle Kmelia, qui sert à la mise en œuvre de ce travail, les liaisons se font au niveau des services, définis par une signature avec types, des assertions (pré/post-conditions) et un système de transition. La partie contrat *CCM*, très simplifiée ici, est exploitée en section 3, la partie vérification *VCCM* en section 4.

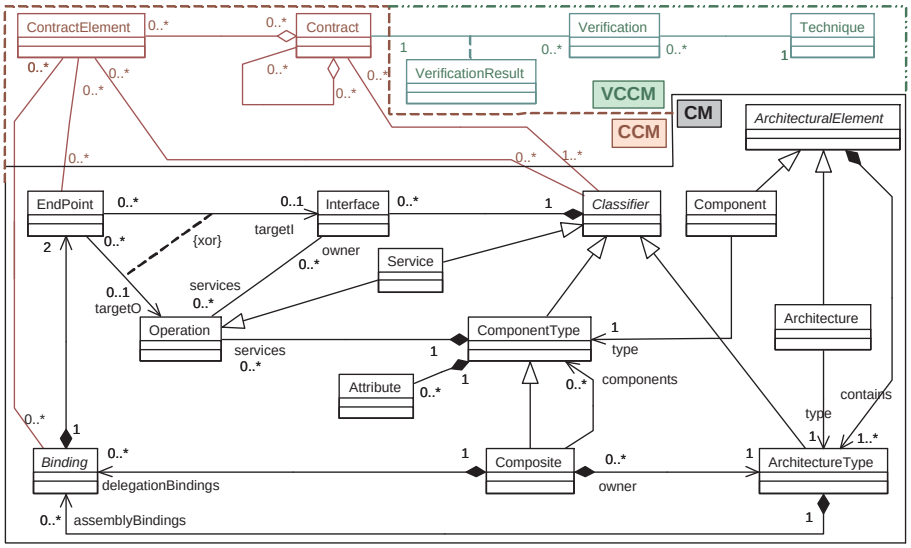

Figure 1. *Métamodèle simplifié d'un modèle à composants, contrats et vérifications*

3. Contrats logiciels en couche sur les éléments de modélisation

La notion de contrat est usuelle pour établir une relation de service entre *classifiers* (*e.g.* classes, composants), perçus comme des boîtes noires encapsulant une implantation. Elle se confond parfois avec les assertions de la conception par contrats, mais ces dernières ne couvrent que les exigences fonctionnelles. La définition des contrats va au-delà lorsqu'elle prend en compte les correspondances de structure, de comportement, les contraintes de communication et de synchronisation, la composition ou la qualité de services (Beugnard *et al.,* 1999, Messabihi *et al.,* 2010). Le tableau ci-après résume le croisement des éléments de modélisation et des niveaux de contrat retenus, structurés en couches d'exigence croissante.

	Éléments de modélisation			
Contrats	**Service**	**Composant**	**Assemblage**	**Composite**
Structurel	correction du typage	cohérence de l'interface	compatibilité de signatures et de structure, disponibilité de services	idem assemblage + observabilité
Fonctionnel	correction fonctionnelle	préservation de l'invariant	compatibilité d'assertions	conformité d'assertions
Dynamique	cohérence comportementale	cohérence du protocole	compatibilité comportementale	conformité comportementale

Le **contrat de service** regroupe la correction de signature, la *correction fonctionnelle* (un service réalise ce qu'il est censé faire) à base axiomatique de Hoare sur

les actions du service et la *cohérence comportementale* (pas d'états incohérents). Le **contrat de composant** prévoit essentiellement l'accessibilité des services (chaîne de dépendance), la cohérence du composant (préservation de son invariant) ainsi que la correction du protocole (enchaînement licite des appels de services). Le **contrat d'assemblage** est un contrat de clientèle classique mais qui se décline sur trois niveaux de compatibilité. Enfin, le **contrat de composite** est une variante du précédent pour la relation parent/enfant (visibilité, promotion). La déclinaison en couche des contrats permet à la fois le ciblage vers des outils spécifiques mais aussi la répercussion des résultats vers les couches de plus bas niveau pour éviter des vérifications inutiles.

4. Mise en œuvre des vérifications

Les contrats étant multi-niveaux et composites, plusieurs vérifications leurs sont associées. De plus, plusieurs techniques peuvent être utilisées pour vérifier un même contrat élémentaire. Les techniques de vérification de contrats sont caractérisées par leur degré d'automatisation, leur complexité et leur applicabilité (qui peuvent être calculées en fonction du modèle) ainsi que la force du résultat (une preuve de propriété est plus forte que l'absence d'échecs lors d'un test, par exemple). Une vérification effectuée produit un résultat, partiel ou complet, assorti d'un diagnostic exploitable par d'autres vérifications. Prenons l'exemple de la vérification du contrat d'assemblage au niveau communication en visant le *model checker* MEC (figure 2). Les vérifications

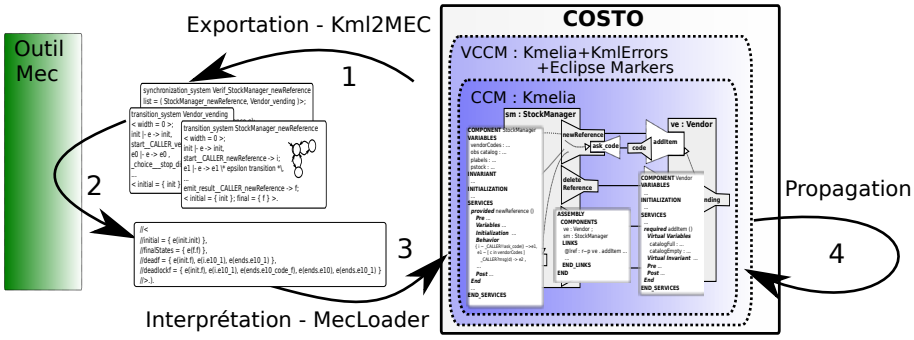

Figure 2. *Transformations associées à une vérification de contrat dans COSTO*

de contrats sont obtenues par une séquence composée d'un calcul opéré par un outil externe et de trois transformations de modèles pour assurer le passage d'un modèle (V)CCM à un modèle vérifié VCCM. L'exportation (1) transforme la partie du modèle nécessaire à la vérification dans le langage de l'outil utilisé. La vérification (2) est réalisée par l'outil, qui produit un résultat en rapport avec le modèle fourni. L'interprétation (3) retranscrit les résultats de l'outil dans le modèle VCCM. La propagation (4) répercute les résultats et les diagnostics sur les contrats du modèle VCCM qui dépendent de la validité du contrat vérifié en (2).

Dans la mise en œuvre, les aspects structurels et de typage sont pour l'essentiel vérifiés dans COSTO[2] mais des transformations du système de transitions dans des outils de graphes sont en cours de développement. Pour la vérification de propriétés liées à la dynamique la plate-forme COSTO comprend aussi une transformation `Kml2Lotos` qui à l'instar de `kml2Mec` dans la figure 2, permet de vérifier le contrat d'assemblage au niveau communication en utilisant la plate-forme CADP. Les exigences fonctionnelles sont toutes vérifiées sous la forme d'obligations de preuves obtenues en ciblant B et les outils AtelierB et Rodin par une série de transformations `Kml2B`. Les tests et l'animation de modèles font appel à la transformation `Kml2Java`. Enfin la transformation `Kml2Latex` génère de la documentation des spécifications et des illustrations au format `dot`. Ces transformations sont des plugins Eclipse de COSTO implantés via des *templates*, des patrons visiteurs et des opérateurs de conversion *ad hoc*.

5. Conclusion

Cet article met en évidence l'intérêt des transformations de modèles pour vérifier des propriétés liées aux contrats avant le déploiement des composants. L'originalité du travail réside dans l'utilisation de vérifications tirant parti de la déclinaison du modèle de contrat en couches étendu à l'ensemble des *classifiers*. Les transformations s'appliquent ainsi à des vues spécifiques aux propriétés, permettant leur vérification dans le support adéquat. Une expérimentation intégrant le métamodèle de composants dans EMF a été menée afin d'exprimer Kmelia dans ce cadre et d'y exprimer les transformations *model to model* pour des supports implantés sous Eclipse, notamment Rodin. Une autre perspective concerne les propriétés des transformations, en particulier préserver les contrats dans le raffinement vers le code via par exemple des contrats de transformation (Cariou *et al.,* 2009). Enfin nous souhaitons lier l'ensemble dans des processus de vérification incluant les chaînes de dépendances et de propagation.

6. Bibliographie

André P., Ardourel G., « Domain Based Verification for UML Models », *in* , L. Kuzniarz, , G. Reggio, , J.-L. Sourrouille, , M. Staron (eds), *Workshop on Consistency in Model Driven Engineering C@Mode'05*, p. 47-62, November, 2005.

Beugnard A., Jézéquel J.-M., Plouzeau N., Watkins D., « Making Components Contract Aware », *Computer*, vol. 32, n° 7, p. 38-45, 1999.

Cariou E., Belloir N., Barbier F., « Contrats de transformations pour la validation de raffinement de modèles », *5èmes journées sur l'Ingénierie Dirigée par les Modèles*, IDM 09, 2009.

Messabihi M., André P., Attiogbé C., « Multilevel Contracts for Trusted Components », *in* , J. Cámara, , C. Canal, , G. Salaün (eds), *WCSI*, vol. 37 of *EPTCS*, p. 71-85, 2010.

Meyer B., « The Grand Challenge of Trusted Components », *Proceedings of 25th International Conference on Software Engineering*, IEEE Computer Society, p. 660-667, 2003.

2. `http://www.lina.sciences.univ-nantes.fr/coloss/download/posterCosto.pdf`